PRESCRIPTION BASED MAINTENANCE MANAGEMENT SYSTEM

G. Scott Valentine
James E. Dzakowic
Impact Technologies, LLC
200 Canal View Blvd
Rochester, NY 14623
scott.valentine@impact-tek.com

Thomas Galie
NSWC
Carderock Division
Philadelphia Naval Business Center,
Philadelphia, PA, 19112-5083

John Scharschan
NSWC
5001 South Broad Street
Philadelphia, PA 19112

Abstract: In recent years, significant focus has been placed on the development and implementation of advanced prognostic and health management (PHM) technologies in military and industrial applications. The term PHM encompasses anomaly, diagnostic and prognostic algorithms as well as higher level reasoning algorithms for isolating root causes of faults/failures and directing optimal operational or maintenance actions. In such systems, two current deficiencies exist. First, for a variety of reasons, component and subsystem interactions in such systems are poorly realized. The issue manifests itself as multiple dependent "boxes" indicating faults with shotgun tests or valuable domain expertise required to de-conflict and reduce ambiguity groups. Secondly, complex systems still largely rely on expert rule-bases for reasoning which are notoriously difficult to maintain over a life cycle and are prone to logical conflicts. This paper begins to address these deficiencies by outlining a simulation-based process for automatically 1) realizing complex system interactions for optimal PHM system design and 2) building and maintaining model-based reasoning architectures where decisions and conclusions naturally precipitate out of a more manageable system model.

Keywords: Condition Monitoring; Open System Architecture; Optimal Maintenance Decisions; Prognostic and Health Management

Introduction: Prognostic and health management embraces a philosophy that works to maximize a piece of equipment's availability to perform a task or mission, while simultaneously minimizing its maintenance cost. In recent years many PHM packages have entered the market incorporating near real time automated monitoring, data analysis, diagnostic routines and more advanced, embedded algorithms driven prognostic applications predicting remaining useful life. The plethora of information derived throughout these multiple layers all garner inputs for decision support tools. These inputs give maintenance planners the knowledge to take appropriate actions. In light of the data now made available to maintainer's, the next natural progression of questions to be asked is "What action should be taken first when sensors may be indicating that several available maintenance actions could correct the failure at hand?" A reasoning engine that learns from its previous experiences and provides guidance on the most likely next course of action is a decision support tool that will compliment the PHM philosophy.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2007		**00-00-2007 to 00-00-2007**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Prescription Based Maintenance Management System	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Impact Technologies LLC,200 Canal View Blvd,Rochester,NY,14618**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**15**	

Prescription-Based Maintenance Management System (PBMS) is such a maintenance reasoner engine that directs the performance of maintenance tasks only when there is sufficient evidence of a need. Towards that, a system of monitoring, detection, diagnosis and prognosis will guide the users to a probable set of maintenance tasks to correct an occurring discrepancy. The next appropriate step will be to prescribe the most likely corrective action, "prescription," to remedy the situation. This process flow is displayed in Figure 1 below. An effective Prescription-Based Maintenance Management System will ultimately reduce the overall operation and support (O&S) costs of shipboard systems, through the efficient use of manpower, equipment and spare parts.

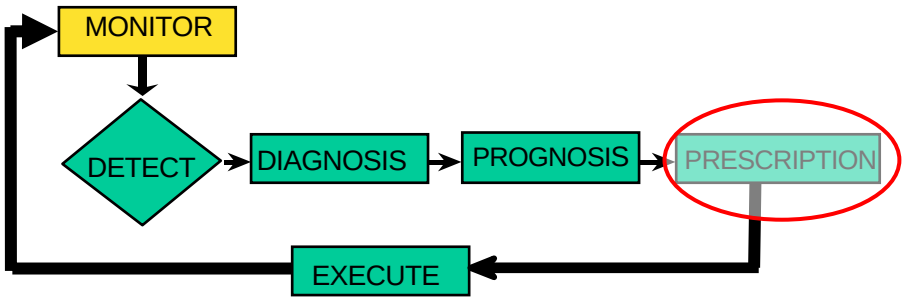

Figure 1: Prescription-Based Maintenance Concept

Overall System Architecture: As with any support tool emphasis must be placed on creating a system architecture that has scalable attributes and that easily communicates with other systems through an open architecture environment. The PBMS application interacts with existing Navy systems such as the Integrated Condition Assessment System (ICAS), existing diagnostic or prognostic agents, scheduling software, electronic instructions, and maintenance databases. Open system standards are used where interaction with other software modules is expected. Evolving Open Systems Architecture for Enterprise Application Integration (OSA-EAI) standards is used for undefined external interfaces where compatibility is required. Capabilities and provisions are being provided for future interfacing with a mission readiness assessment system. The PBMS external interfaces are outlined in Figure 2.

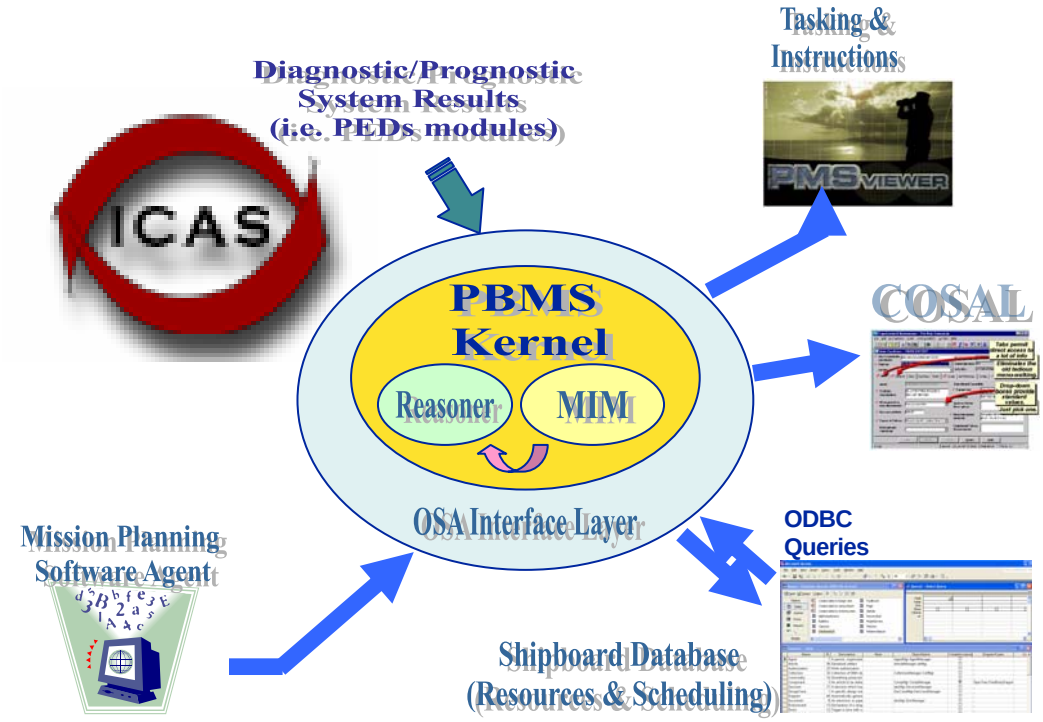

Figure 2: PBMS Data/Info Integration

During a SBIR Phase II program, Impact Technologies, LLC developed the PBMS shell in the Tcl/Tk programming language displayed with XML for web base capability. Tcl/Tk is an extremely versatile high-level language with a large pallet of Graphical User Interface (GUI) elements and Open System Architecture object links such as COM, DCOM, CORBA and XML. The XML modeling approach is used to model and analyze the reasoner-based system. The developer can use standard notation to model system components, behaviors, and users. This approach is widely accepted for distributed object-oriented systems. This architecture enables systems in dynamic link library (DLL) files to interface with the evaluation system through a standardized OSA protocol. An XML-based Maintenance Integrated Model (MIM) will be integrated into the core application program. The MIM incorporates a fault tree type database with the appropriate condition monitors and maintenance tasks in a graphical portal.

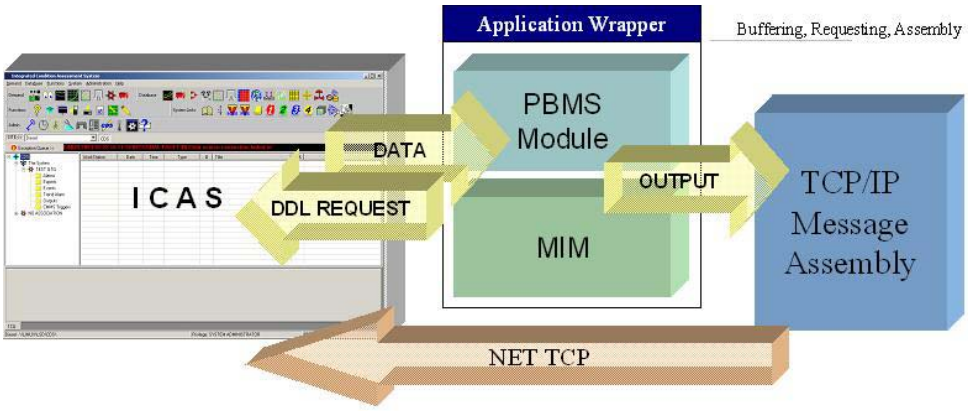

Figure 3: PBMS Integration with ICAS

As an integral part of the PBMS, information from diagnostic or prognostic agents is utilized to trigger appropriate maintenance actions. Machinery health assessments from diagnostic or prognostic agents are passed through ICAS to the PBMS as failure mode evidence sources for maintenance reasoning. Other classes of information and data such as operating mode are also being exchanged as required. The PBMS application wrapper is interfaced between the ICAS application and TCP/IP messaging service, Figure 3, where a data demand function pulls the appropriate data and information from ICAS to the PBMS application. The PBMS module itself directly interfaces with the MIM that has previously captured the knowledge associated with the CBM technologies and maintenance tasking. The overall system architecture envelops three distinct agents: Maintenance integrated model, PBMS Reasoner, and Reinforced Learning.

Maintenance Integrated Model (MIM): The Maintenance Integrated Model incorporates a fault tree type database with the appropriate condition monitors and maintenance tasks in a graphical portal. The majority of the information in the MIM should come from system Failure Modes, Effects, and Criticality Analysis (FMECA). The FMECA information that is required to build a complete MIM model is listed below.

- Functional Elements (Systems, Subsystems, Components) and hierarchy
- Relationships between functional elements (Mechanical, Electrical, Fluid)
- Failure Modes or Faults
    - Failure Rate (1/MTTF – related to Weibull Parameters)
    - Weibull Shape Parameter
    - Severity (1:Negligible, 2:Marginal, 3:Critical, 4:Catastrophic)
- Symptoms and Effects
- Relationships between Failure Modes and Effects (Fault Tree)
- Probability of Propagation between each element of the fault tree
- Monitors or Algorithms (Built in Test or Discrete, Diagnostics, Prognostics)
    - Probability of Detection for BIT or Diagnostic
    - Probability of False Alarm for BIT or Diagnostic
    - For Prognostics, use Accuracy
    - Algorithm Life Cycle Cost
- Sensors that are required for the Algorithms
    - Sensor failure rate

    - Sensor Life Cycle Cost
- Relationships between Monitors and Symptoms/Effects or Failure Modes
- Maintenance Tasks that mitigate failure modes (Unit, Intermediate, Depot)
    - Hardware cost to perform task
    - Labor cost to perform task
    - Hours of downtime required to perform task
    - Unscheduled task penalty factor
    - Maintenance Task Effectiveness
- Connectivity between Maintenance Task and Failure Mode
-

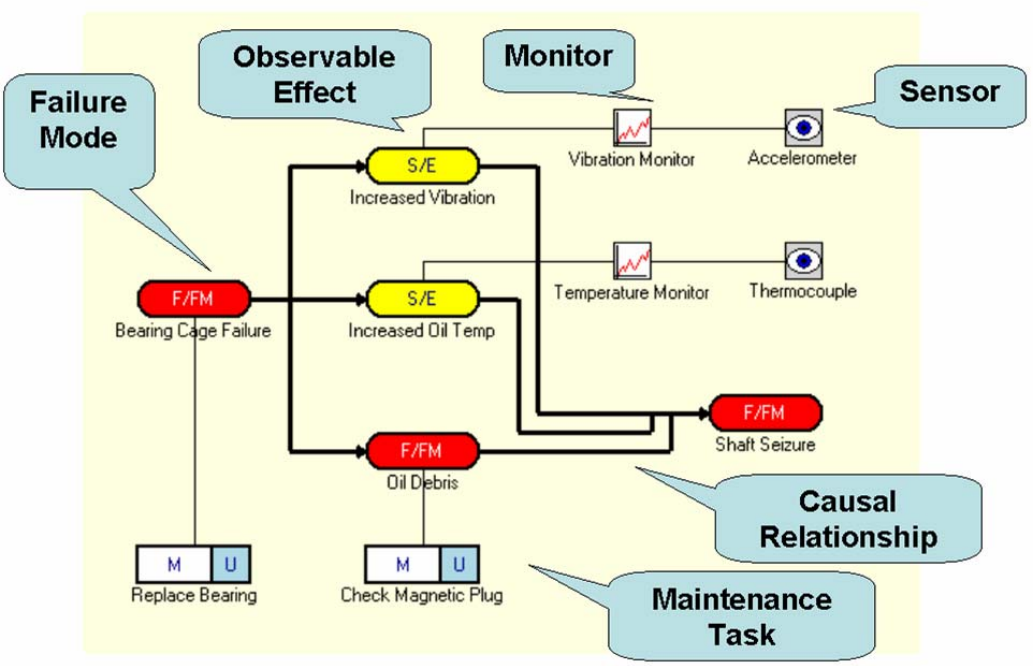

Figure 4: Example MIM with annotations

All MIM information is stored in specially formatted XML database files. As needed, any of the layers may access the file to extract pertinent information. The XML files contain two types of elements – components and connections. Components refer to systems, subsystems, system elements, faults and failure modes, effects, monitors, sensors, and maintenance tasks. Each component instance has a variety of associated attributes. Connections can refer to mechanical, electrical, and fluid links between system elements, or associations between various health management components (faults and failure modes, effects, monitors, sensors, and maintenance tasks). The structured XML approach allows standardized code to be used to read and interpret the data.

PBMS Reasoners: The reasoning algorithm being developed under this task represents the logical process through which decisions are made by the PBMS. In this case, a Reasoning algorithm must be able to identify root cause failure modes and corrective actions (maintenance tasks) based on all available evidence, information/knowledge sources and a practical set of restraints.

Most present day maintenance practices involve an experienced person to troubleshoot an observed problem with a system and decide upon the appropriate maintenance action. Currently used Interactive Electronic Tech Manuals (IETM) step the maintainers through a series of tests / procedures on the basis of convenience, cost or, at best, likelihood of success. The most modern IETM have "meta-tags" which act as handles for indexing to certain tasks. If something in the IETM is incorrect, it remains incorrect until a human changes it. The suggested maintenance tasking is static and most importantly, human interpretation is required to characterize and classify evidence. The automated reasoning algorithms and agent-based approaches demonstrated in this application can circumvent these issues of classification and characterization of sometimes incongruent data, to make a logical progression from failure modes to most appropriate maintenance tasks.

Evidence-Based Maintenance Reasoning: In the Phase II program, Impact Technologies is developing a knowledge-based fusion algorithm that is capable of performing failure mode and maintenance task ranking, as well as root cause isolation within a system containing a variety of information and knowledge sources already mentioned. While novel in its integration capabilities, the technique is based on fundamental mathematical principles and can significantly improve upon the performance of the results from the techniques previously introduced. A standard FMECA of the system provides the basic requirement of information for the proposed Impact Reasoner implementation.

The Impact Fusion Reasoner (IFR) will have two modes under which it receives/processes information to act upon: Initialization and Evidence Activation. Essentially, Reasoner information calculated in the initialization mode will be based on static model attributes such as evidence to failure mode dependencies, false alarm rates and real fault probabilities. During initialization, two steps will be required before reasoning can actually take place: Connectivity Matrix generation (CM) and Evidence Sub Graph (ESG) identification. The CM is generated by exploring the relationships between failure modes and monitors configured in the model. Figure 5 illustrates an example where Boolean numbers are used to generate the connectivity (note: the term "monitor" is equated with any piece of evidence). A value of "1" indicates that a piece of evidence is able to directly observe the effects of a specific failure mode. No direct observation corresponds to a value of "0".

Connectivity Matrix				
	Failure Mode 1	Failure Mode 2	Failure Mode 3	Failure Mode 4
Monitor 1	1	0	0	1
Monitor 2	0	0	1	0
Monitor 3	0	1	0	1

Figure 5: Connectivity Matrix

The Evidence Sub Graph (ESG) can be extracted from the Connectivity Matrix. An ESG contains generic monitors (i.e. evidence) that are directly or indirectly related to the same failure modes in the system. Two generic monitors observing a fault would be considered indirectly related through a failure mode while the relation between a monitor and an observable fault is direct.

The Reasoning Process Invoked Upon Monitor Activation: The Reasoning process is broken into three major steps: Time-based (temporal) considerations, Failure Mode Ranking and Failure Rate Adjustment. Each monitor could have time delays associated to its responsiveness to fault detection, as well as failure propagation paths with associated monitors that activate in a temporal sequence. As an example, consider the fact that an oil leak may only be identified after a prescribed amount of oil has been lost. Accounting for delays is an important feature that insures that all potential evidence that may be used to indict a root failure mode has been considered before a conclusion is reached (see Figure 6 as an example).

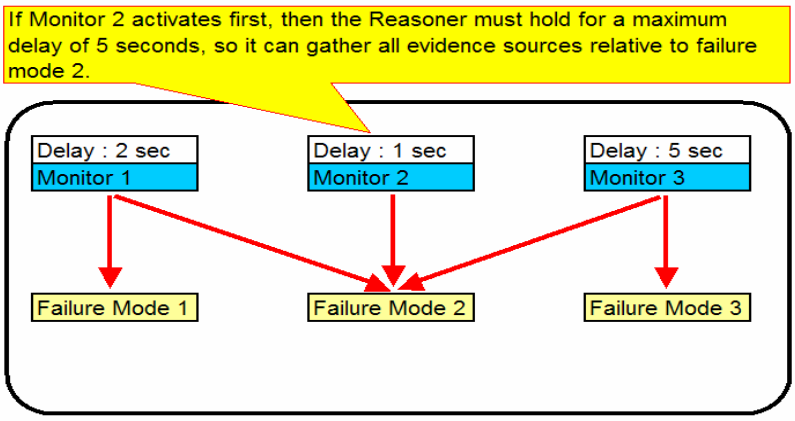

Figure 6: Reasoning with monitor time delays

Accounting for Positive and Negative Evidence: Positive and Negative Evidence, along with Real Fault Probabilities and False Alarm Rates need to directly contribute to the failure mode ranking process. Positive Evidence may be defined as evidence that indicts a failure mode consistent with the evidence sub-graph (ESG). An absence of evidence when it is expected may be defined as Negative Evidence. Figure 7 illustrates these concepts for three Monitors capable of observing a single Failure Mode in a generic example. Consider Failure Mode 1 to be a lube oil leak for a generic piece of machinery. One could characterize at least three pieces of evidence that, when combined, should indict this problem 1) loss of pressure 2) loss of oil level 3) observation of oil on the ground. If evidence 1 and 2 are present but not 3, there is imperfect evidence indicting the lube oil leak failure mode. It is entirely possible; however, that the oil may be evaporating, the leak wasn't found or that the pressure and level readings are inaccurate.

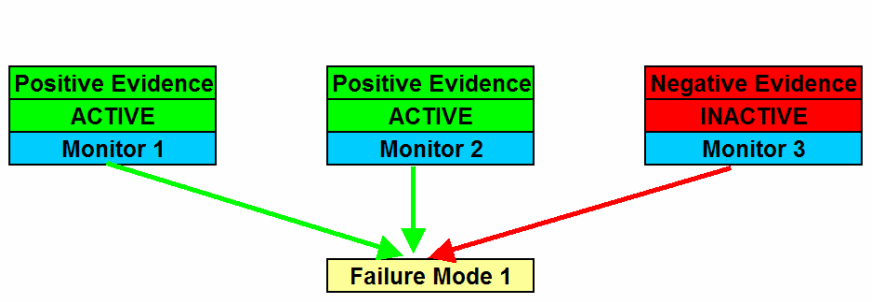

Figure 7: Positive and Negative Evidence Sources

Real Fault Probabilities and False Alarm Rates: The probability of a fault existing given that a monitor has indicted it is (P (FM/Monitor)) and corresponds to the Real Fault Probability (RFP). The False Alarm Rate (FAR) is related to the potential of a monitor activating given the absence of a fault (P (Monitor/~FM)). Equation 1 illustrates how the ranking of a given failure mode is determined through the consideration of Positive evidence, Negative evidence, FAR and RFP. This is a fundamental equation of the Impact Reasoner.

$$Rank_{FM(i)} = 1 + \frac{\sum_{j=1}^{PosE}(1 - FAR_j) - \sum_{k=1}^{NegE} RFP_k}{(PosE) + (NegE)} \qquad (1)$$

The Positive Evidence contribution is reduced by 1 minus the potential of a monitor going active without the presence of a fault (False Alarm Rates). Negative Evidence contribution is reduced by 1 minus the potential of a fault existing but going undetected (1-Real Fault Probability). The numerator is normalized by the sum of the total potential of the system. The Rank ranges from 0 to 1, with the highest number being the best score. A second step is required to complete the ranking process, which addresses failure rates and can be used to break "ties" on ranking.

Accounting for failure rates: The initial ranking algorithm alone will not account for a-priori failure rates. For example, if the 2 failure modes have equal initial ranks, consideration of the historical failure rate of one relative to the other can act as the tie-breaker. The premise in the failure rate adjustment process is that, while failure rate statistics have no place in the initial ranking they can be appropriately mapped into a ranking adjustment of the potential set of root cause failure modes.

The ranking adjustment can be performed through a predefined look up chart, which converts the original rank to an adjusted rank based on failure rate values. The chart is a function of initialization variables, failure rates and original ranks. The initialization algorithm provides crucial parameters necessary in mapping the charts dynamic nature. Such a chart is illustrated in Figure 8. The values on the abscissa represent the original ranking values based upon Positive and Negative evidence. The ordinate includes two separate axes: a failure rate log-scale axis and a scale factor axis. The failure rate values are converted into a scale factor, which defines the y-intercept of a specific line. The log to linear scale conversion is based on the fact the failure rate are usually orders of

magnitude apart (i.e. 1E-6 vs. 1E-5). The chart contains a mapping of two main regions separated by a variable ranking limit.

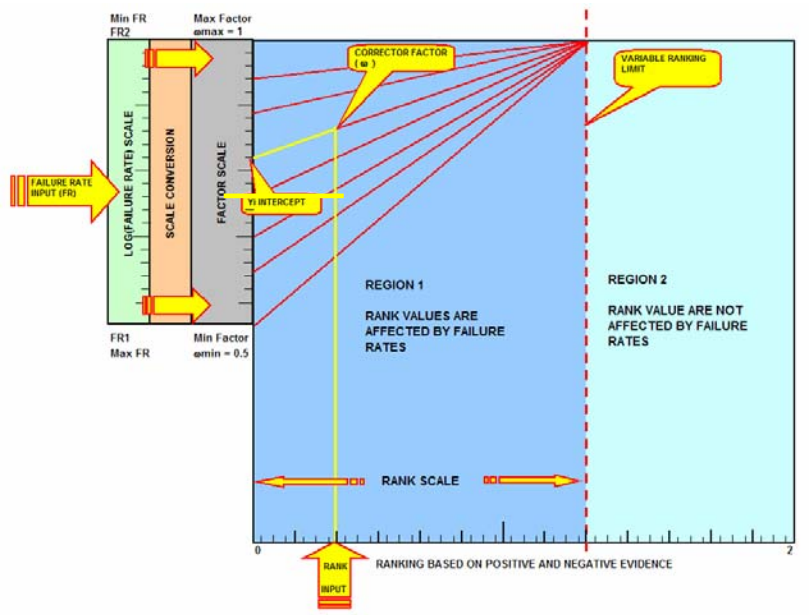

Figure 8: Failure Rate Adjustment Chart

The ranking limit can be adjusted along the abscissa changing the distribution of the set of lines and subsequently affecting the two regions. Failure rates will be irrelevant for any rank values that fall to the right of the limit. Any values to the left of the limit will be affected. The adjusted rank for a given failure mode is a calculated ratio of the original rank and a factor number provided by the chart as shown in Equation 2.

$$Adj\_Rank_{FM(i)} = \frac{Rank_{FM(i)}}{\omega_{FM(i)}} \tag{2}$$

The derivation of the correction factor requires the y-intercept and the slope of the line. Interpolating between two axes defined on the ordinate outputs a y-intercept measure on the factor scale axes.

$$Y_i = \left[\frac{(\omega_{\min} - 1)}{(\log(MaxFR) - \log(MinFR))} \cdot (\log(FR) - \log(MinFR)) \right] + \omega_{\max} \tag{3}$$

The slope of the line is calculated from Equation 4. The Rank Scale ranges between the origin and the ranking limit.

$$Slope = \frac{MaxFactor - Y_i}{Rankscale} \tag{4}$$

The corrector factor is calculated from Equation 5 and substituted into Equation 2. Using the original rank input, the slope of the line and its y-intercept the corrector factor is calculated and substituted into Equation 2.

$$\omega_{FM(i)} = (Rank_{FM(i)} \cdot Slope) + Y_i \tag{5}$$

The ranking adjustment process as stated will only allow failure mode rankings to improve based on failure rates.

Reinforced Learning: Q-Learning is among a series of Reinforcement Learning (RL) algorithms with a common objective aimed to successfully solve problems that require complex decision processing. Reinforcement Learning is a combination of fields such as dynamic programming and supervised learning, which lead to powerful intelligent machine learning systems. One of the biggest advantages of these techniques is the model-free aspect. The applicability of reinforcement learning algorithms is independent from the structure of the system taken into consideration.
Q-Learning has been integrated in the MIM environment to reinforce the maintenance task decision processes and address the evolvable aspect of the paradigm. Suppose a maintainer is presented with a problem or symptom which leads to the choice of multiple solutions that may not necessarily lead to an effective maintenance action. No matter which solution the maintainer decides to perform, the reinforcement learning system records these problem/solutions relationships and their outcome. During the course of time the RL system will ultimately lead the maintainer to choose the most effective solution to the given problem.
The following relationships formally introduce the reinforcement learning model.

- S : A discrete set of environment states
- A : A discrete set of agent actions
- R : A set of scalar reinforcement signals

An agent is connected to its environment through actions. For every interaction, an agent receives an indication of the current state, s, of the environment. Once the agent knows the current probable state, it chooses among a set of actions. The selected action ‘a’ will consequently change the state ‘s’ of the environment. During the transition from the current state to the new state a scalar reinforcement signal ‘r’ is used to reward or penalize the action based on its outcome. After numerous interactions and transitions the agent will gain useful experience that will lead to choosing the optimal action ‘a’ for a given current state ‘s’. Once the system has learned over time, a policy can then be used to choose which actions should be performed when a state is encountered. The value of the state is defined as the sum of the reinforcements over a discrete number of interactions. The optimal policy can be calculated by indexing the mapping between state and actions that maximize the sum of the reinforcements.

The advantage of Q-learning is that initially, when the optimal state to action relationships are not entirely known, the agent may have the option of randomly selecting an action. Q-learning allows arbitrary experimentation and simultaneously preserves the current best estimate state values. The final results in Q-learning are not contaminated by experimental actions, which will only ultimately reward or penalize the state action pairs. In order to obtain meaningful values, it is necessary for the agent to try out each action in every state multiple times allowing the reinforcement updates to assign penalties or rewards based on the outcome.
The Q-learning equation is presented below:

$$Q(s_t, a_t) = (1-\alpha)Q(s_t, a_t) + \alpha(r(s_t, a_t) + \gamma \max Q(s_{t+1}, a\ )) \qquad (6)$$

Where, Q is defined as the sum of all reinforcements for the state action pair (s, a) at a given time (t).

- α is the learning rate parameter [0,1]
- γ is the discount parameter [0,1]

The learning rate parameter α carries a significant importance in adjusting the algorithms' learning behavior. α controls the weighting on current and previous knowledge represented in the state-action pair values. As the learning rate parameter approaches 1, new experience outweighs previous experience. As it approaches 0, previous experience outweighs new experience. For the state – action pairs, representing task effectiveness of a maintenance action relative to a fault, α is set closer to 0, outweighing previous experience to new, which translates in assigning more importance to the maintenance task history performance.

The discount parameter γ controls the weighting on the new state action pair. For the current application, the parameter is fixed to a constant and provides scaling to the state-pair values, which does not affect the overall results of the Q-Learning algorithm.
The mapping between state and action pairs can be represented in an n x m matrix where

- n represents the number of possible states
- m represents the number of possible actions

Q - Matrix				
s1	Q(s1,a1)	Q(s1,a2)	Q(s1,a3)	Q(s1,a4)
s2	Q(s2,a1)	Q(s2,a2)	Q(s2,a3)	Q(s2,a4)
s3	Q(s3,a1)	Q(s3,a2)	Q(s3,a3)	Q(s3,a4)
	a1	a2	a3	a4

Figure 9: Q Learning Matrix

The Q-Matrix may be a representation of the Task Effectiveness of a particular maintenance action (a) directly connected to a failure mode (s). The outcome of the

performed action leads to a state transition. The state transitions are defined in the following two cases and illustrated in Figure 10 below.

- Case 1 : Given a Failure Mode (S), if any particular selected action is successful, the state will transition from the current fault state to a healthy state.

- Case 2 : Given a Failure Mode (S), if any particular selected action is not successful, the state will transition from the current fault state back to its current fault state or to a different fault state. The transition to a different faulty state could be an indication that the failure mode was not clearly identified until after the maintenance action was performed.

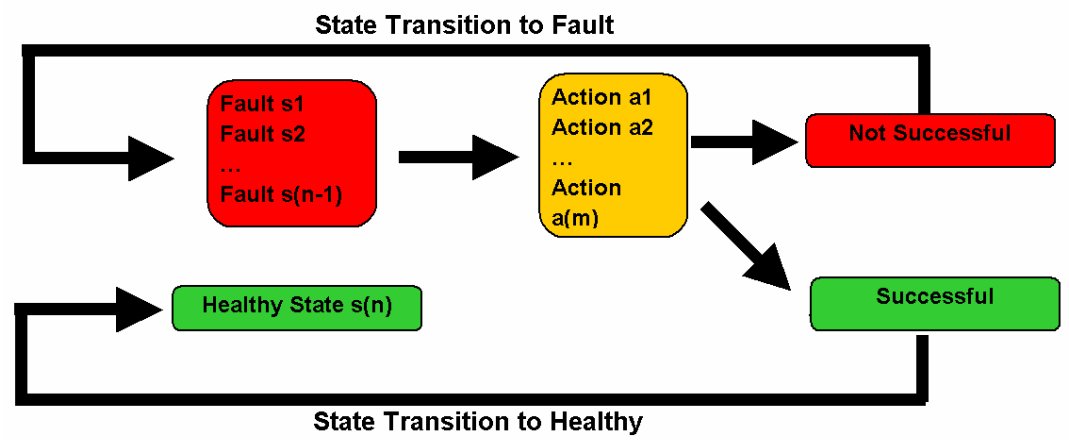

Figure 10: Decision Process Flow Chart

Implementation of the Application: The overall PBMS software package will be used to enhance existing CBM processes currently used for naval applications. The PBMS offers "plug and play" capability with software packages such as ICAS and enables the ability to intelligently recommend maintenance actions based on health management evidence, logistics, economics, and other factors. The design of the PBMS software architecture is flexible in that it may receive both archived and real-time data. Because it will be used to assess real-time information, the domain logic was designed to minimize response time. The reasoner also relies on close interaction with the MIM to perform its function.

The reasoner layer takes in evidence sources in the form of CBM monitor activations, processes the evidence in conjunction with a MIM, and outputs a list of probable failure modes and ranked maintenance task recommendations. The overall maintenance ranking can be modified to account for criteria weighting factors, availability, and scheduling. The current suite of ranking criteria includes: hardware and labor cost, availability or downtime considerations, safety, and criticality. Considerations such as common procedures, part/tool/labor availability, and upcoming scheduled tasks are factored into the domain logic.

The Human System Interface (HSI) provides a real-time link between the outputs of the PBMS (the system) and its user (the human). It is represented by a Graphical User Interface (GUI) that will present the information to the user in a manner that is easily understood as seen if Figure 11.

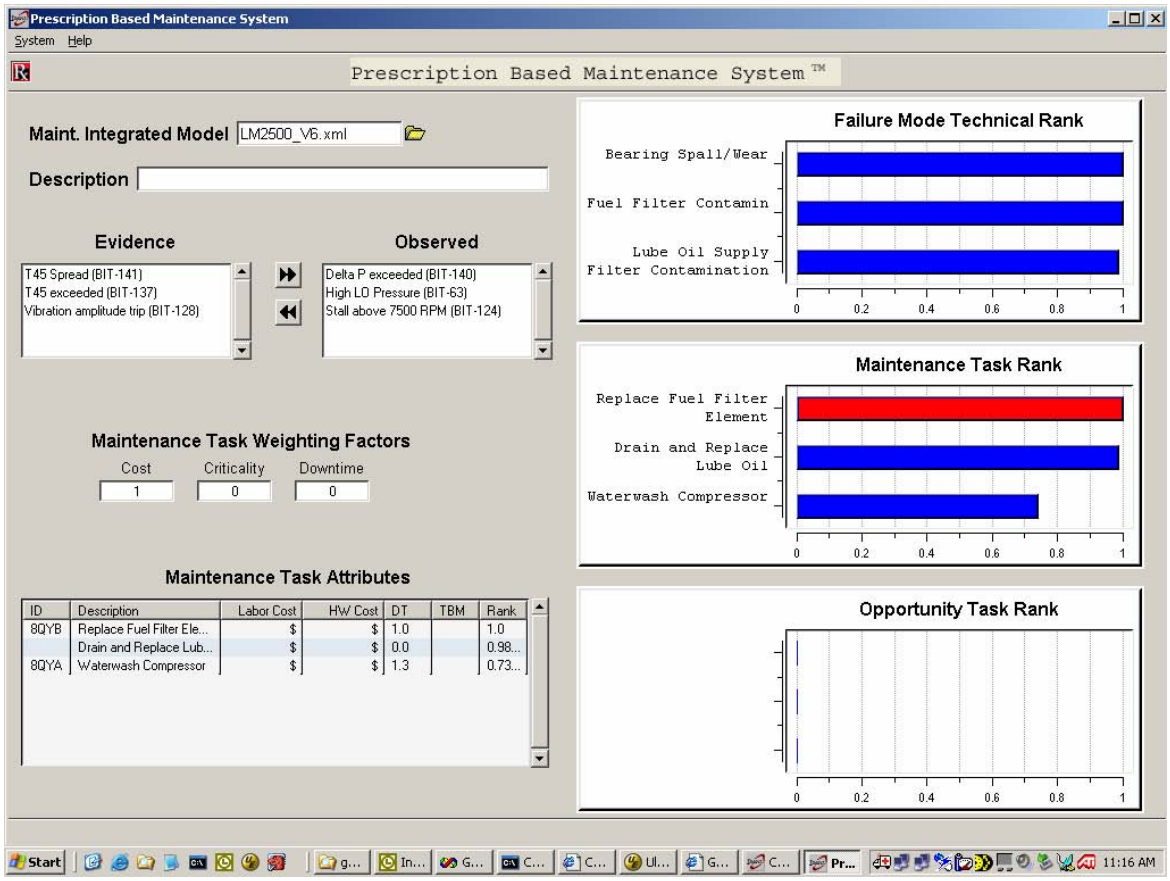

Figure 11: PBMS Graphical User interface

The HSI presents the maintenance technician with recommended maintenance actions and accepts maintainer feedback. Work order generation capabilities have been incorporated into the design based on the PMS Viewer format. Feedback provided by the user shall be used to update failure mode and maintenance task rankings, and if necessary update the MIM to make it more effective. Specifically, the PBMS reasoner must know which maintenance tasks were performed and whether or not they were successful.

The ship's information systems contain data that are updated in near real time and can be utilized by the PBMS. The Integrated Condition Assessment System (ICAS) contains event, fault and Built-in-Test (BIT) data. This evidential data is input to the PBMS and activates evidence when a fault condition is detected. Typically, the evidence trigger is a BIT or diagnostic, but several prognostic enhancements are available that will submit Remaining Useful Life (RUL) data as evidence to the PBMS.

The PBMS reasoner takes in the evidence provided by ICAS and, utilizing a subsystem's predefined Maintenance Integrated Model (MIM), determines the most likely failure modes and consequential maintenance tasks required to correct the fault. Once the maintenance tasks have been defined, the reasoner queries the Consolidated Onboard Ships Supply data base to determine whether or not the appropriate tools, parts, materials and personnel are available to complete the maintenance task. The reasoner flags the tasks that are missing resources so the human-in-the-loop can make a more informed decision for maintenance scheduling.

The final component of the PBMS is the activation of a learning mechanism through maintainer feedback. The feedback system is used to determine if a prescribed maintenance task was successful at correcting a fault condition. The link between failure modes and maintenance tasks is accordingly strengthen or weakened based on the success of the task or lack thereof.

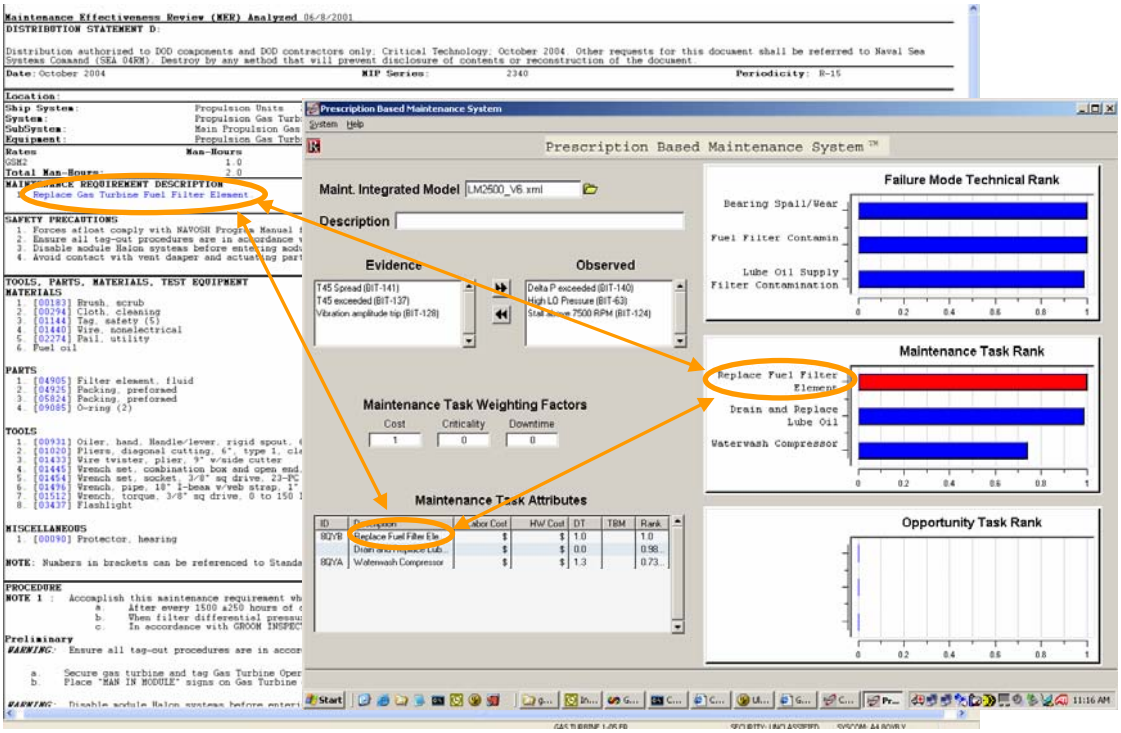

Figure 12: Connectivity between PBMS & PMS Viewer

Conclusions: A comprehensive approach has been presented to guide maintainers to the optimal maintenance actions as diagnosis through automated Condition-based maintenance systems. This integrated approach operates in open system architecture, using accepted OSA standards, to easily communicate with disparate data systems. The PBMS reasoners implement a variety of techniques such as Fusion reasoning Real fault probabilities, False alarm rates and advanced Reinforce learning. Finally, a human system interface concept was presented for illustrating how information from a complicated health management system could be presented to an end user.

References:

1. Bonarini, “Evolutionary Learning, Reinforcement Learning, and Fuzzy rules for Knowledge Acquisition in Agent Based Systems”, Proceedings of IEEE, Vol. 89, No. 9, September 2001

2. Sakai, Ikeda, Iwata, “A New Criterion using Information gained for Action Selection Strategy in Reinforced Learning”, IEEE Transaction on Neural Networks, Vol. 15, No. 4, July 2004

3. Kacprzynski, Palladino, “Self-Evolving Maintenance Knowledge Bases”, SBIR Phase I Final report N68335-04-C-0181, November 2004